Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2003-40

A MEMETIC HEURISTIC FOR THE GENERALIZED QUADRATIC ASSIGNMENT PROBLEM

by

Jean-François Cordeau
Manlio Gaudioso
Gilbert Laporte
Luigi Moccia

December 2003

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone : (514) 343-7575 / Télécopieur : (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

PUBLICATION
CRT-2003-40

A MEMETIC HEURISTIC FOR THE GENERALIZED QUADRATIC ASSIGNMENT PROBLEM

by

Jean-François Cordeau[1]
Manlio Gaudioso[2]
Gilbert Laporte[1]
Luigi Moccia[1,2]

This research was supported in part by the Canada Research Chair in Distribution Management, by the HEC Montréal Centre for International Business Studies, and by the CN Chair in Transport Economics and Intermodality.

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7, and Canada Research Chair in Distribution Management, HEC Montréal, 3000, Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[2] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2003

104-02002/

Abstract

In the *Generalized Quadratic Assignment Problem* (GQAP) we are given n weighted facilities, m capacitated sites, a traffic intensity matrix between facilities, a distance matrix between sites, unit traffic costs, and assignment costs of facilities to sites. The aim is to determine an assignment of facilities to sites in such a way that the sum of assignment and traffic costs is minimized and the total weight of all facilities assigned to the same site does not exceed the site capacity. The GQAP is a generalization of the *Quadratic Assignment Problem* (QAP) in which $n = m$ and exactly one facility must be assigned to each site. The problem has applications in container yard management and in the assignment of equipment to manufacturing sites. This article describes a memetic heuristic for the GQAP, as well as an integer linear programming formulation which can be solved by CPLEX for small size instances. For larger instances, feasible solutions can be obtained by means of a truncated branch-and-bound procedure. Computational experiments show that on small instances the proposed heuristic always yields an optimal solution; on larger instances it always outperforms the truncated branch-and-bound algorithm.

Keywords: *generalized quadratic assignment problem; memetic heuristic; genetic algorithm; tabu search.*

Résumé

Dans le problème d'affectation quadratique généralisé (PAQG), on considère un ensemble de n installations, un ensemble de m sites dotés d'une capacité, une matrice décrivant l'intensité des mouvements entre les installations, des coûts unitaires de transport et des coûts fixes d'affectation des installations aux sites. Le problème est de déterminer une affectation des installations aux sites de telle manière que la somme des coûts d'affectation et de transport soit minimisée et que la capacité totale requise par les installations affectées à un site ne dépasse pas la capacité de ce site. Le PAQG est une généralisation du problème d'affectation quadratique dans lequel $n = m$ et exactement une installation doit être affectée à chaque site. Le problème a des applications dans la gestion de cours de conteneurs et dans l'affectation d'équipements à des sites de production manufacturière. Cet article décrit une heuristique mémétique pour le PAQG ainsi qu'une formulation de programmation linéaire en nombres entiers qui peut être résolue par CPLEX pour des instances de petite taille. Pour les plus grandes instances, des solutions réalisables peuvent être obtenues en utilisant une méthode de séparation et évaluation partielle. Des expériences de calcul montrent que pour les petites instances, l'heuristique trouve toujours une solution optimale; sur les plus grandes instances, elle donne toujours de meilleurs résultats qu'une méthode de séparation et évaluation partielle.

Mots clefs: *problème d'affectation quadratique généralisé; heuristique mémétique; algorithme génétique; recherche avec tabous.*

1 Introduction

In the *Generalized Quadratic Assignment Problem* (GQAP) we are given n weighted facilities, m capacitated sites, a traffic intensity matrix between facilities, a distance matrix between sites, unit traffic costs, and assignment costs of facilities to sites. The aim is to determine an assignment of facilities to sites in such a way that the sum of assignment and traffic costs is minimized and the total weight of all facilities assigned to the same site does not exceed the site capacity. The GQAP was introduced by Lee and Ma (2003). It generalizes the *Quadratic Assignment Problem* (QAP) in which $n = m$ and exactly one facility must be assigned to each site, and it is therefore NP-hard (Garey, and Johnson, 1979).

An application of the GQAP arises in the management of container yards where the yard is partitioned into storage areas, corresponding to sites, and groups of containers correspond to facilities. The containers are grouped according to their inbound and outbound *port routes*. A port route is the sequence of ports visited by a vessel and is determined by the shipping company. Because shipping companies often own storage areas for their different port routes, containers must be moved between areas if the inbound and outbound port routes differ, which generates traffic costs. The problem consists of locating the container groups in the storage areas in order to minimize the yard handling movements (Gaudioso et al., 1999 and 2001). Another application described by Lee and Ma (2003) relates to an equipment location problem in a Canadian based manufacturing company. Sets of equipments must be assigned to manufacturing sites and intermediate parts move between these sites. Transportation is carried out by an outside firm payed on the basis of travel and waiting time. Because the overall transportation cost is high, the company wishes to optimally locate its equipments in its different manufacturing sites.

1.1 Literature review

The scientific literature on the GQAP is rather limited. We are aware of only one scientific article (Lee and Ma, 2003) on this problem. It provides Integer Linear Programming (ILP) formulations of the problem together with a branch-and-bound algorithm consisting of two phases. In a first phase, a feasible solution is constructed by means of a greedy algorithm that operates on the GQAP objective function and considers three initial orderings for sites and four initial orderings for facilities, yielding a total of 12 combinations. A second version of the greedy algorithm works with assignment costs only. The best solution value found by all versions of the greedy algorithm is used as an initial upper bound in a second phase where facilities are gradually assigned to sites in a branch-and-bound fashion. At each node of the search tree a lower bound on the objective function value is computed by taking into account the costs associated with the sets of already assigned and yet unassigned facilities. This enumeration algorithm was capable of optimally solving instances including up to 16 facilities and seven sites.

The QAP literature is also relevant to the study of the GQAP. The QAP was introduced by Koopmans and Beckman in 1957 and remains to this day a very difficult combinatorial optimization problem. Several benchmark instances of size $n = 36$ introduced by Steinberg in 1961, and instances of size $n = 30$ defined by Nugent et al. in 1968, have only recently been solved to optimality. For recent articles on exact algorithms for the QAP, see Burkard et al. (1998), Rendl (2002), and Anstricher (2003). A large number of heuristic methods have been also proposed for the QAP. The most successful are simulated annealing (Connoly, 1990), tabu search (Taillard, 1991), hybrid genetic-tabu search (Fleurent and Ferland, 1994), GRASP (Li et al., 1994), ant colonies (Gambardella et al., 1999) and, more recently, a hybrid genetic proposed by Drezner (2003a, b) which combines a new genetic merging procedure followed by a post-optimization phase based on variable neighborhood search (Mladenović and Hansen, 1997).

1.2 Outline of the remainder of this article

Apart from the greedy heuristic of Lee and Ma (2003), we are aware of no other approximate algorithm for the GQAP. Our aim is to devise an accurate and fast memetic heuristic for this problem. The GQAP is formulated in Section 2 as a Quadratic Integer Program which is then linearized. The memetic heuristic is introduced in Section 3, followed by computational results in Section 4, and by conclusions in Section 5.

2 Mathematical model

The following sets and variables are used to formulate the GQAP:

- $N = \{1, ..., n\}$, a set of facilities;
- $M = \{1, ..., m)$, a set of sites;
- $x_{ik} \in \{0, 1\}$, facility $i \in N$ is assigned to site $k \in M$ if and only if $x_{ik} = 1$.

The input data for the problem are:

- t_{ij}, the traffic intensity between facilities $i \in N$ and $j \in N$;
- d_{hk}, the distance between sites $h \in M$ and $k \in M$; we assume that distances are symmetric, i.e., $d_{hk} = d_{kh}\ \forall h, k$;
- a_{ik}, the cost of assigning facility $i \in N$ to site $k \in M$;
- v, the unit traffic cost;
- q_i, the weight of facility $i \in N$;
- Q_k, the capacity of site $k \in M$.

The formulation of the GQAP is then:

$$\text{minimize} \sum_{i \in N} \sum_{k \in M} a_{ik} x_{ik} + v \sum_{i \in N} \sum_{j \in N} \sum_{h \in M} \sum_{k \in M} t_{ij} d_{hk} x_{ih} x_{jk} \tag{1}$$

subject to

$$\sum_{k \in M} x_{ik} = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{i \in N} q_i x_{ik} \leq Q_k \qquad \forall k \in M, \tag{3}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in N, \forall k \in M. \tag{4}$$

In this formulation the objective function (1) minimizes the sum of assignment and traffic costs. Constraints (2) impose that every facility is assigned to one and only one site, and constraints (3) ensure the capacity of each site is not exceeded.

Lee and Ma (2003) have introduced the following linearization for the GQAP, based on QAP linearizations proposed by Frieze and Yadegar (1983) and by Padberg and Rijal (1996). This linearization, refereed to as LGQAP, will be used to assess the quality of our heuristic on small size instances, where optimality is obtained within acceptable computation time; we also use the best solution provided by a branch-and-bound resolution of LGQAP to compute an upper bound on the optimal value of medium size instances. LGQAP uses additional sets and variables:

- $N(i) = \{j \in N : j > i\}$;
- z_{ikjh}: equal to 1 if and only if facility i is assigned to site k and facility j is assigned to site h.

The linearization is as follows:

$$\text{minimize} \sum_{i \in N} \sum_{k \in M} a_{ik} x_{ik} + v \sum_{i \in N \setminus \{n\}} \sum_{k \in M} \sum_{j \in N(i)} \sum_{h \in M} (t_{ij} + t_{ji}) d_{hk} z_{ikjh} \tag{5}$$

subject to

$$\sum_{k \in M} x_{ik} = 1 \qquad \forall i \in N, \tag{6}$$

$$\sum_{i \in N} q_i x_{ik} \leq Q_k \qquad \forall k \in M, \tag{7}$$

$$\sum_{h \in M} z_{ikjh} = x_{ik} \qquad \forall i \in N \setminus \{n\}, \forall k \in M, \forall j \in N(i), \tag{8}$$

$$\sum_{k \in M} z_{ikjh} = x_{jh} \qquad \forall i \in N \setminus \{n\}, \forall j \in N(i), \forall h \in M, \tag{9}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in N, \forall k \in M, \tag{10}$$

$$0 \leq z_{ikjh} \leq 1 \qquad \forall i \in N \setminus \{n\}, \forall k \in M, \forall j \in N(i), \forall h \in M. \tag{11}$$

In the computational experiments carried out by Lee and Ma (2003) a comparison was made between three linearizations of the GQAP implemented in CPLEX, and a new specialized branch-and-bound algorithm. These experiments showed that no approach dominated the others. We decided to use the LGQAP formulation as a benchmark because it yielded shorter average computational times in the Lee and Ma experiments.

3 Heuristic

We now describe the components of our heuristic which combines genetic search (Holland, 1975) and tabu search (Glover, 1986). The integration of these two meta-heuristics is sometimes referred to as a memetic algorithm (Moscato and Cotta, 2003). The algorithm randomly generates an initial *population* P of feasible solutions. A cycle of genetic iterations is then applied to P. At each iteration a merging procedure creates an offspring which is then improved trough at most η iterations of tabu search. If the resulting improved offspring is better than the worst member of the population and does not already belong to P, it then replaces the worst solution; otherwise the offspring is discarded. To enhance the exploration of different areas of the solution space, a multi-start approach is also used, meaning that the algorithm is started ϑ times and the best solution is retained. We now present the components of the heuristic in more detail.

3.1 Initial population

Denote by S the set of solutions satisfying constraints (2), (3), and (4). A first solution $s \in S$ is randomly generated by assigning facilities to sites while attempting to respect the capacity constraints. Reaching feasibility is not always possible, particularly in the case of tightly constrained instances. If s is infeasible, an attempt to reach feasibility is made through the application of the TABUFIX search procedure described in Section 3.4. If TABUFIX fails, new solutions are generated until a desired population size $|P|$ is reached.

3.2 Genetic procedure

The genetic procedure attempts to improve the population set P by merging, at each iteration ω, two solutions to create an offspring. The merging rule used is inspired by the work of Drezner (2003a, b) for the QAP. Our problem is different from that of Drezner since it contains capacity constraints. As a result the merging procedure we have developed for the GQAP is new.

For each site $k \in M$ the median value of the distances between the site k and the other $m-1$ sites is first computed. Sites whose distance from k is less than the median value are called *cohesive* with respect to k. All sites that are at the same distance as the median value are called *neutral*, and the remaining sites are called *non-cohesive*. For each site k it is then possible to partition M into a subset M_1^k of cohesive sites, a subset M_2^k of neutral sites, and a subset M_3^k of non-cohesive sites. Site k is considered as belonging to M_1^k. A site k selected to partition M into subsets is called *a pivot*. The construction of the partitions of M for each pivot is performed only once at the beginning of the algorithm. In Figure 1 we provide a simple example with eight sites equally spaced along a line. The three corresponding subsets for four sites chosen as pivots are illustrated using different shades.

The merging procedure randomly selects two solutions of the population pool called parents. The parent A_1 having the smaller solution value is called *first parent* and the other parent A_2 is called *second parent*. Ties are broken arbitrarily. Like all elements of P, these two solutions satisfy the capacity constraints. The merging procedure operates as a cycle of iterations applied to all sites $k \in M$. Each iteration attempts to generate a new solution s^k. In s^k a site h belonging to the cohesive subset M_1^k is assigned all the facilities previously assigned to that site h in the first parent A_1. Similarly a site in the non-cohesive subset M_3^k is assigned all facilities from the second parent A_2, if these facilities are not already assigned to the cohesive subset. After this phase some facilities may still be unassigned. An attempt to assign them randomly to sites, while preserving feasibility, is then made. If no feasible assignment can be reached the solution is discarded. If a feasible solution s^k is obtained, it is then compared to the best known solution s_ω^* found at iteration ω among those corresponding to all pivot sites; if s^k improves upon s_ω^*, it then replaces it.

Merging two parents can be unsuccessful if no feasible solution is encountered during the m iterations. In this case another couple is randomly selected and the process is repeated until

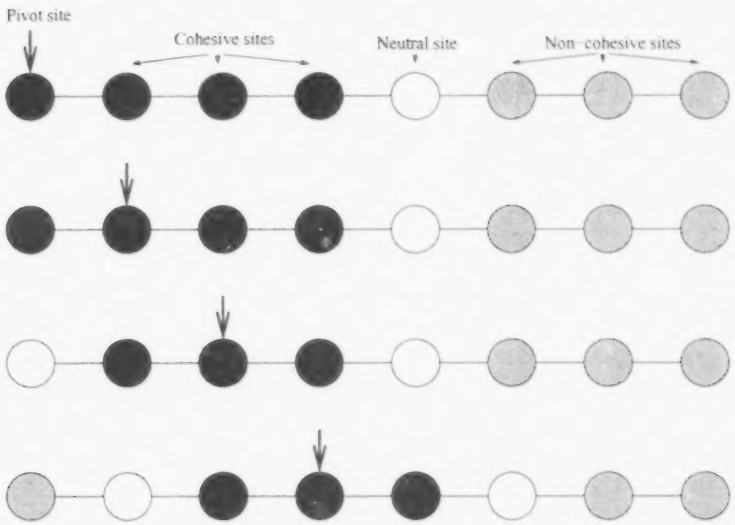

Figure 1: Example of partition construction induced by different pivot choices.

an offspring is generated. After every successful or unsuccessful merging, the two parents are labeled as *forbidden*, and no other merging is attempted with them. This feature is not really necessary when the maximum number ϖ of genetic iterations is small compared to population size since the probability of a repeated merger is small. However, when ϖ is large and the merging success rate is low, then this feature yields a noticeable improvement in the performance of the merging procedure.

We now illustrate one iteration of the merging cycle using the graph of Figure 1. Eleven facilities must be assigned to eight sites equally spaced along a line. The weight q_i of facility $i \in \{1, ..., 11\}$ is equal to one if i is odd and equal to two if i is even. The capacity Q_k of each site k is equal to two. Figure 2 shows, from top to bottom, the site partition induced by the choice of the pivot site $k = 1$, the assignment corresponding to the first parent, the partial assignment representing the offspring, and the assignment corresponding to the second parent. The offspring is obtained by first taking from the first parent the assignments of the cohesive set M_1^1 and then the yet unassigned facilities belonging to the non-cohesive set of the second parent. The three facilities 4, 6 and 8 are unassigned at this stage. Because any attempt to assign these to the unsaturated sites 5, 6, 7 and 8 will fail, no feasible solution exists. Therefore the offspring generated with the pivot site $k = 1$ is discarded. Note that in this example the capacity constraints are tight since $\sum_k Q_k = \sum_i q_i$.

After the merging procedure a post-optimization tabu search heuristic (POTS) is called to improve the best offspring s^*_ω. The resulting solution is compared with the worst solution in P and is discarded if it does not improve upon it. Otherwise the improved offspring is compared with the other solutions in P. If the offspring is not original, meaning that a similar solution

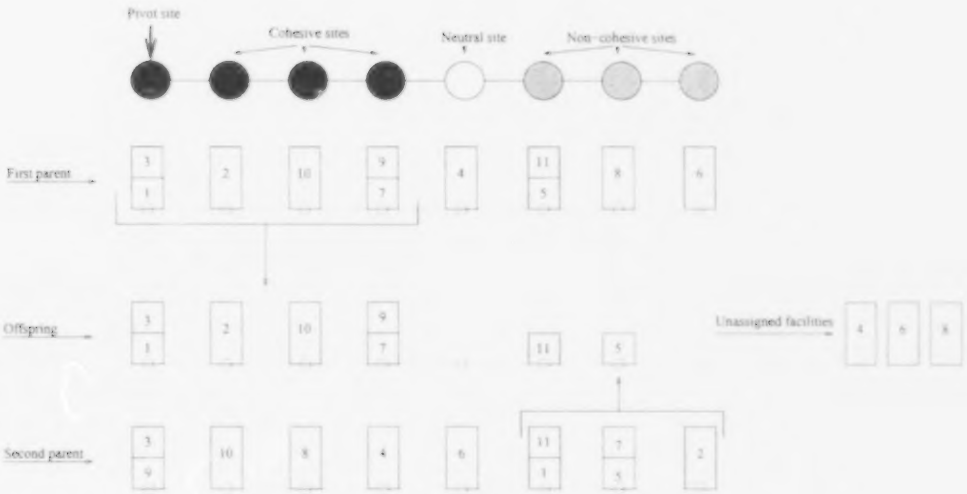

Figure 2: Example of one iteration of the merging cycle.

already exists in P, it is discarded; otherwise, it is inserted in the population P to replace the worst solution.

At the end of an iteration the genetic iteration counter ω is incremented and another iteration is performed until a limit ϖ is reached. The genetic cycle may terminate prematurely if the difference between the number of new and non-new solutions generated exceed a limit ψ proportional to $|P|$. This means that population P becomes exhausted in terms of genetic diversification and a new random population is required. Whenever $\omega > \varpi$ or the premature termination occurs the genetic procedure gives the control to the multi-start cycle which compares the best solution found during the current genetic cycle with the preceding ones, if any.

The genetic procedure can be summarized as follows:

- for $\omega = 1, ..., \varpi$ { /* genetic cycle */
  - Do {
    - randomly select a non-forbidden couple A_1 and A_2 in P
    - for $k = 1, ..., m$ { /* merging cycle */
      - merge A_1 and A_2 using k as pivot and obtain s^k, if feasible
      - compare s^k to the best known offspring s^*_ω, if any}
  - } until an offspring is generated
  - Apply POTS to the offspring s^*_ω which eventually is improved
  - Compare s^*_ω with the worst and the best solutions in P

- If s^*_ω is better than the worst and not better than the best solution, check whether it is original
- Replace the worst solution with s^*_ω if it is original and better than the worst solution
- Update the counter for original and non-original solutions and eventually decide to prematurely terminate the genetic cycle}

3.3 Post-optimization tabu search

The POTS procedure allows temporary violations of the capacity constraints. Denote by S^A the set of solutions satisfying the assignment constraints (2) and (4) but possibly violating the capacity constraints (3). Let $c(s)$ denote the cost of solution $s \in S^A$, and let $w(s)$ be the total violation of capacity constraints. Solutions are then evaluated by means of a penalized cost function $f(s) = c(s) + \alpha w(s)$, where α is a positive parameter. By dynamically adjusting the value of this parameter the relaxation mechanism facilitates the exploration of the search space and is particularly useful to escape local minima. Parameter α is updated at each iteration as follows: if the new current solution s is feasible with respect to the capacity constraints (3), then α is divided by a factor $1 + \delta$, where δ is a positive parameter; otherwise α is multiplied by $1 + \delta$. For tightly constrained instances this scheme may not be sufficient to drive the search process toward a feasible solution. Therefore, when α becomes larger than a preset value α^+ the TABUFIX procedure (Section 3.4) is called to regain feasibility.

The heuristic explores the solution space by moving at each iteration from the current solution $s \in S^A$ to the best solution in its neighborhood $N(s)$. The tabu search method is based on the definition of attributes used to characterize the solutions of S^A. These attributes are also used to control tabu tenures and to implement a diversification strategy. With each solution $s \in S^A$ is associated an attribute set $B(s) = \{(i,k)$: facility i is assigned to site $k\}$. The neighborhood $N(s)$ of a solution s is defined by applying a simple operator that removes an attribute (i,k) from $B(s)$ and replaces it with another attribute (i,k'), where $k \neq k'$. When facility i is removed from site k, its reassignment to k is declared forbidden for the next θ iterations by assigning a tabu status to attribute (i,k). An aspiration criterion allows the revocation of the tabu status of an attribute if that would allow the search process to reach a solution of smaller cost than that of the best solution identified possessing that attribute.

To diversify the search, any solution $\overline{s} \in N(s)$ such that $f(\overline{s}) \geq f(s)$ is penalized by a factor proportional to the addition frequency of its distinguishing attribute, and by a scaling factor. More precisely, let ρ_{ik} be the number of times attribute (i,k) has been added to the solution during the process and let ζ be the number of the current iteration. If attribute (i,k) is added to s to obtain $\overline{s}$, a penalty $p(\overline{s}) = \lambda c(\overline{s})\rho_{ik}/\zeta$ is added to $f(\overline{s})$, where $p(\overline{s}) = 0$ if $f(\overline{s}) < f(s)$. The scaling factor $c(\overline{s})$ introduces a correction to adjust the penalties with respect to the total solution cost. Finally, a parameter λ is used to control the intensity of the diversification. These penalties have the effect of driving the search process toward less explored regions of the search space.

At the end of this procedure POTS is restarted and the iteration counter ζ is incremented to keep track of the the number of iterations performed by TABUFIX. Every time an improved feasible solution is reached a counter ζ^* is set to ζ, the number of the current iteration. When the difference $\zeta - \zeta^*$ becomes larger than a preset value μ, POTS terminates and the current best solution is returned to the genetic cycle. The accelerated termination rule, before $\zeta = \eta$, has proved very effective in reducing the computational time on some instances.

3.4 Recovering feasibility: procedure TABUFIX

TABUFIX is an auxiliary procedure called by the population procedure and POTS when a feasible solution cannot be reached. It starts from an infeasible solution s with positive violation $w(s)$ of the capacity constraints. The cost function $f(s) = c(s) + \alpha w(s)$ is replaced with $f(s) = w(s)$ and thus the penalization mechanism is no longer necessary. The neighborhood structure and all the other tabu search mechanisms are the same as those of POTS. However TABUFIX ends as soon as a feasible solution is reached, meaning that $w(s) = 0$. The maximum number of iterations allowed is set equal to n^2m. When the procedure ends it returns the number of iterations performed to the calling procedure.

4 Computational results

The procedures just described were coded in ANSI C and the LGQAP formulation was implemented in CPLEX 8.1. Computational experiments were performed on a SUN workstation (1.2 GHz). We first describe how test instances were generated, and we then provide results obtained with our heuristic compared with those obtained with the LGQAP.

4.1 Test instances

We have conducted tests on two set of instances, I1 and I2. The set I1 is made up of the 26 instances used by Lee and Ma (2003). These instances are labeled with a code consisting of four parts: n, m, and two letters defined as follow:

- E, if the facilities have a wide range of space requirements;
- F, if the facilities have roughly the same space requirements;
- G, if the sites have a wide range of capacities;
- H, if the sites have roughly the same capacities.

The traffic intensities, distances, assignment and travel costs are the same for each combination of E, F, G, H for a given instance of size n and m. It turns out that these instances are not very challenging for our heuristic. We have therefore designed a new set of 21 larger and more difficult instances, called I2. They are labeled with three letters: n, m, and a parameter

$f \in \{1, ..., 100\}$ which controls the tightness of the capacity constraints. For given n, m and f, the input data were generated as follows:

- t_{ij} and d_{ij} were generated according to a discrete uniform distribution in $[1, ..., 100]$, both matrices are symmetric;
- a_{ik} was generated according to a discrete uniform distribution in $[1, ..., 10^4 n]$;
- v was generated according to a discrete uniform distribution in $[1, ..., 5]$;
- q_i was generated according to a discrete uniform distribution in $[1, ..., 100]$;
- Q_k was generated according to a discrete uniform distribution in $[1, ..., \sum_{i \in N} q_i / fm]$.

Trivial infeasible instances were discarded by means of additional controls.

4.2 Implementation details and results

The parameters used in the various procedures are as follows:

- ψ: limit when the genetic procedure is terminated prematurely, equal to 100;
- θ : tabu duration, equal to $\lfloor n \log m \rfloor$;
- λ : diversification intensity parameter, equal to 0.015;
- δ : penalty adjustment parameter, equal to 0.05;
- α^+: maximum value of the penalty before the activation of TABUFIX, equal to 1.5×10^4;
- μ: limit when POTS is terminated prematurely, equal to 100.

The computation time of the heuristic is influenced by four parameters: $|P|$ population size, ϑ number of times the algorithm is started, ϖ maximum number of genetic iterations, η maximum number of POTS iterations.

For the I1 instances these four parameters were set to 75, 4, 300 and 150 respectively. The instances of set I1 were easily solved by CPLEX and by the heuristic, which was always able to reach the optimum. Table 1 provides the computation times of CPLEX and the heuristic. On this set of instances the CPLEX MIP parameter is set equal to one, which emphasizes feasibility in the branch-and-bound tree exploration. This choice yields the shorter computation times out of the four options, which are:

- MIP = 0, default, balance optimality and feasibility;
- MIP = 1, emphasis on feasibility;
- MIP = 2, emphasis on optimality;
- MIP = 3, best bound.

The more challenging instances are those of subset I2. Here CPLEX could not reach an optimal solution within the prescribed time limit except on one instance (30-08-55). We have therefore used a truncated branch-and-bound which stops with the best feasible solution identified after two hours. Here the choice of the CPLEX MIP parameter has a noticeable impact as illustrated in Table 2. On average the best solutions were obtained with MIP = 2. However, a different choice of the MIP parameter yields better results on some instances. On two instances (30-20-95 and 35-15-95) CPLEX was not able to find a feasible solution within the time limit.

In Table 3 we report results produced by the heuristic with four different parameter sets $\{|P|, \vartheta, \varpi, \eta\}$. The results are presented with increasing computation times from left to right, and the corresponding four parameter sets are:

- PS1: $|P| = 100, \vartheta = 10, \varpi = 3000, \eta = 3000$;
- PS2: $|P| = 100, \vartheta = 30, \varpi = 3000, \eta = 3000$;
- PS3: $|P| = 200, \vartheta = 30, \varpi = 30000, \eta = 3000$;
- PS4: $|P| = 100, \vartheta = 300, \varpi = 30000, \eta = 3000$.

The heuristic exhibits a stable behavior since the quality of the solutions does not improve significantly when longer computational times are allowed.

Table 4 provides a comparison between CPLEX and the heuristic. The best solution values out of the four runs of CPLEX are chosen as reference. The best and the worst values out of the four runs of the heuristic are also reported. The advantages of the heuristic are clear. It yields a feasible solution on instances where CPLEX fails, and the worst solutions produced by the heuristic dominate the CPLEX solutions except for one instance (30-07-75). The worst heuristic solutions correspond to shorter computation times. Given slightly longer computation times the heuristic always produces better solutions than CPLEX does.

4.3 Performance of the heuristic components

To assess the contribution of the two main components of the heuristic, the genetic cycle and the post-optimization tabu search, we ran two additional computational experiments by disabling POTS or the genetic cycle:

- Genetic: POTS is disabled, $\eta = 0$;
- Tabu: the genetic cycle is disabled, $\varpi = 0$.

The non-disabled parameters, (ϑ for the multi-start cycle, ϖ for Genetic, and η for Tabu) were set to match the average computational time of the heuristic on the instance set I1, increased by 50%. The results are reported in Table 5. On all instances, neither Genetic nor Tabu were able to reach an optimal solution. On the average they produced similar results.

Finally, the usefulness of the TABUFIX procedure is illustrated in Table 6 where the computation times of the memetic heuristic are compared with those of a version of the heuristic

in which TABUFIX is disabled. The instances reported in the table correspond to tight capacity constraints for which the random generation of the initial population often fails, yielding longer computation times; when the capacity constraints are loose TABUFIX plays no major role. On one particular instance (12-3-F-H) the heuristic without TABUFIX does not terminate within the allotted time limit of two hours.

5 Conclusions

We have developed a memetic heuristic for the GQAP. To test the heuristic a linearization of the GQAP was implemented in CPLEX. The solution values provided by the heuristic was compared on small instances to the optimal values generated by the exact GQAP formulation. On these instances our heuristic always yields optimal solutions. On larger instances it always outperforms the truncated CPLEX branch-and-bound algorithm applied to the GQAP.

Acknowledgments

The authors thank Zvi Drezner who made his QAP genetic-hybrid computer code available to the authors, and Chi-Guhn Lee who provided the instance set I1. This work was supported by the Canada Research Chair in Distribution Management, by the HEC Montréal Centre for International Business Studies, and by the CN Chair in Transport Economics and Intermodality.

References

[1] K. M. Anstreicher, 2003. Recent Advances in the Solution of Quadratic Assignment Problems. *Mathematical Programming* 97, 27-42.

[2] R. E. Burkard, 1991. Location with Spatial Interactions: the Quadratic Assignment Problem. In *Discrete Location Theory*, P. B. Mirchandani, and R. L. Francis (eds). Wiley, New York, 387-437.

[3] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis, 1998. The Quadratic Assignment Problem. In *Handbook of Combinatorial Optimization*, D.-Z. Du, and P. M. Pardalos (eds). Kluwer, Boston, 241-337.

[4] D. T. Connoly, 1990. An Improved Annealing Scheme for the QAP. *European Journal of Operations Research* 46, 93-100.

[5] Z. Drezner, 2003a. Heuristic Algorithms for the Solution of the Quadratic Assignment Problem. *Journal of Applied Mathematics and Decision Sciences*, in press.

[6] Z. Drezner, 2003b. A New Genetic Algorithm for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, in press.

[7] C. Fleurent, and J. A. Ferland, 1994. Genetic Hybrids for the Quadratic Assignment Problem. *DIMACS Series in Mathematics and Theoretical Computer Science 16*, 190-206.

[8] A. Frieze, and J. Jadegar, 1983. On the Quadratic Assignment Problem. *Discrete Applied Mathematics 5*, 89-98.

[9] L. M. Gambardella, É. D. Taillard, and M. Dorigo, 1999. Ant Colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society 50*, 167-176.

[10] M. Gaudioso, R.-D. Matteo, and G. Morrone, 1999. Sviluppo di un Sistema per l'Analisi e la Clusterizzazione dei Servizi: SACS. Technical Report n. 3/99. Laboratorio di Logistica, Università della Calabria, Italy.

[11] M. Gaudioso, A. Chiodo, and D. Paese, 2001. Un Modello per la Gestione degli Spazi di Piazzale per il Terminale Marittimo Container di Gioia Tauro. Technical Report n. 1/01. Laboratorio di Logistica, Università della Calabria, Italy.

[12] M. R. Garey, and D. S. Johnson, 1979. *Computers and Intractability*. Freeman, San Francisco.

[13] F. Glover, 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research 13*, 533-549.

[14] J. H. Holland, 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.

[15] T. C. Koopmans, and M. J. Beckmann, 1957. Assignment Problems and the Location of Economics Activities. *Econometrica 25*, 53-76.

[16] C.-G. Lee, and Z. Ma, 2003. The Generalized Quadratic Assignment Problem. *Discrete Applied Mathematics*, in press.

[17] Y. Li, P. M. Pardalos, and M. G. C. Resende, 1994. A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. *DIMACS Series in Mathematics and Theoretical Computer Science 16*, 237-261.

[18] N. Mladenović, and P. Hansen, 1997. Variable Neighborhood Search. *Computers & Operations Research 24*, 1097-1100.

[19] P. Moscato, and C. Cotta, 2003. A Gentle Introduction to Memetic Algorithms. In *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger (eds). Kluwer, Boston, 105-144.

[20] C. E. Nugent, T. E. Vollman, and J. Ruml, 1968. An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. *Operations Research 16*, 150-173.

[21] M. W. Padberg, and M. P. Rijal, 1996. *Location, Scheduling, Design and Integer Programming*. Kluwer, Boston.

[22] F. Rendl, 2002. The Quadratic Assignment Problem. In *Facility Location: Applications and Theory*, Z. Drezner, and H. W. Hamacher (eds). Springer-Verlag, Berlin, 439-457.

[23] L. Steinberg, 1961. The Backboard Wiring Problem: A Placement Algorithm. *SIAM Review 3*, 37-50.

[24] É. D. Taillard, 1991. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing 17*, 443-455.

Instance code	CPLEX (sec.)	Heuristic (sec.)
10-5-E-G	3	2
10-5-E-H	7	2
10-5-F-G	4	2
10-5-F-H	7	2
12-3-E-G	1	1
12-3-E-H	2	2
12-3-F-G	2	2
12-3-F-H	3	2
12-4-E-G	3	5
12-4-E-H	13	2
12-4-F-G	16	2
12-4-F-H	43	2
12-5-E-G	13	2
12-5-E-H	17	2
12-6-E-G	30	3
12-8-E-G	41	4
14-6-E-G	23	5
14-7-E-G	91	5
14-8-E-G	65	6
14-9-E-G	199	7
15-6-E-G	76	5
15-7-E-G	75	6
15-8-E-G	244	7
16-6-E-G	112	6
16-7-E-G	1020	7
5-24-E-G	1	2
5-30-E-G	2	2
Average	*78*	*4*

Table 1: Computational results for instance set I1.

Instance code	MIP = 0	MIP = 1	MIP = 2	MIP = 3	Minimum
20-15-35	1614656	**1500497**	1548575	1716379	**1500497**
20-15-55	1802162	1781889	**1770270**	1869404	**1770270**
20-15-75	1989271	**1970971**	2028124	2031770	**1970971**
30-06-95	5268356	**5217536**	5235841	5399200	**5217536**
30-07-75	4395470	4417834	**4383923**	4572073	**4383923**
30-08-55	**3501695**	**3501695**	**3501695**	**3501695**	**3501695**
30-10-65	3675880	3675880	**3662738**	4213402	**3662738**
30-20-35	**3852509**	4224378	**3852509**	3855306	**3852509**
30-20-55	**3915632**	4238537	**3915632**	3954509	**3915632**
30-20-75	4680398	4657075	4680398	**4324822**	**4324822**
30-20-95	n.a.	n.a.	n.a.	n.a.	n.a.
35-15-35	4825595	**4561029**	4825595	4829500	**4561029**
35-15-55	**4850270**	5183585	**4850270**	4920750	**4850270**
35-15-75	**7065340**	7387542	**7065340**	7181838	**7065340**
35-15-95	n.a.	n.a.	n.a.	n.a.	n.a.
40-07-75	7783862	7783862	**7683214**	7783862	**7683214**
40-09-95	**8184249**	**8184249**	**8184249**	9389262	**8184249**
40-10-65	**9482857**	**9482857**	**9482857**	**9482857**	**9482857**
50-10-65	10817535	11073198	**10785572**	11316983	**10785572**
50-10-75	**12073994**	13100779	**12073994**	12717638	**12073994**
50-10-95	**15820477**	**15820477**	**15820477**	16013278	**15820477**
Average	*6084221*	*6198098*	*6071120*	*6267080*	**6031979**

Table 2: Computational results for CPLEX on instance set I2 with different values of the MIP parameter. Bold entries correspond to the minimum solution value for each row.

Parameter set	PS1		PS2		PS3		PS4		
Instance code	Solution	Time (min.)	Solution	Time (min.)	Solution	Time (min.)	Solution	Time (min.)	Minimum
20-15-35	**1471896**	1.5	**1471896**	4.6	**1471896**	9.4	**1471896**	46.7	**1471896**
20-15-55	**1723638**	1.6	**1723638**	5.2	**1723638**	10.2	**1723638**	51.6	**1723638**
20-15-75	**1953188**	2.1	**1953188**	6.5	**1953188**	12.7	**1953188**	66.2	**1953188**
30-06-95	**5160920**	1.8	**5160920**	5.4	**5160920**	10.6	**5160920**	54.6	**5160920**
30-07-75	4400593	1.6	**4383923**	4.8	**4383923**	9.7	**4383923**	48.4	**4383923**
30-08-55	**3501695**	1.7	**3501695**	5.2	**3501695**	10.4	**3501695**	52.8	**3501695**
30-10-65	3624283	2.8	**3620959**	8.4	**3620959**	16.9	**3620959**	83.4	**3620959**
30-20-35	**3379359**	4.8	**3379359**	14.0	**3379359**	29.7	**3379359**	144.5	**3379359**
30-20-55	**3593105**	5.7	**3593105**	18.4	**3593105**	35.9	**3593105**	182.4	**3593105**
30-20-75	**4050938**	7.8	**4050938**	24.2	**4050938**	48.2	**4050938**	242.5	**4050938**
30-20-95	5732543	22.3	5726530	66.5	5726530	135.7	**5710645**	673.6	**5710645**
35-15-35	4457348	4.8	**4456670**	14.0	**4456670**	29.1	**4456670**	141.8	**4456670**
35-15-55	**4639128**	6.0	**4639128**	18.6	**4639128**	36.5	**4639128**	181.0	**4639128**
35-15-75	**6301723**	7.3	**6301723**	21.6	**6301723**	43.2	**6301723**	213.8	**6301723**
35-15-95	6770566	10.6	6773876	30.0	**6689421**	62.3	6729852	300.9	**6689421**
40-07-75	**7405793**	3.2	**7405793**	9.4	**7405793**	18.9	**7405793**	93.7	**7405793**
40-09-95	7747974	7.6	7736126	20.6	7717099	44.3	**7667719**	209.4	**7667719**
40-10-65	**7265559**	4.7	**7265559**	14.2	**7265559**	28.6	**7265559**	144.9	**7265559**
50-10-65	**10513029**	7.1	**10513029**	22.6	**10513029**	47.3	**10513029**	220.5	**10513029**
50-10-75	11282024	9.2	11254069	27.3	**11217503**	58.5	**11217503**	278.7	**11217503**
50-10-95	12998064	12.3	12918462	36.4	**12845598**	76.1	**12845598**	371.2	**12845598**
Average	*7211569*	*8.3*	*7207250*	*24.5*	*7193246*	*50.4*	*7190762*	*245.8*	**7186719**
	100.35		100.29		100.09		100.06		**100.00**

Table 3: Computational results for the memetic heuristic on instance set I2. Bold entries correspond to the minimum solution value for each row; last row gives the normalized value of the average with respect to the minimum average.

Instance code	Best CPLEX	Best Memetic	100(C-M)/M	Worst Memetic	100(C-M)/M
20-15-35	1500497	**1471896**	1.94%	**1471896**	1.94%
20-15-55	1770270	**1723638**	2.71%	**1723638**	2.71%
20-15-75	1970971	**1953188**	0.91%	**1953188**	0.91%
30-06-95	5217536	**5160920**	1.10%	**5160920**	1.10%
30-07-75	**4383923**	**4383923**	0.00%	4400593	-0.38%
30-08-55	**3501695**	**3501695**	0.00%	**3501695**	0.00%
30-10-65	3662738	**3620959**	1.15%	3624283	1.06%
30-20-35	3852509	**3379359**	14.00%	**3379359**	14.00%
30-20-55	3915632	**3593105**	8.98%	**3593105**	8.98%
30-20-75	4324822	**4050938**	6.76%	**4050938**	6.76%
30-20-95	n.a.	**5710645**	n.a.	5732543	n.a
35-15-35	4561029	**4456670**	2.34%	4457348	2.33%
35-15-55	4850270	**4639128**	4.55%	**4639128**	4.55%
35-15-75	7065340	**6301723**	12.12%	**6301723**	12.12%
35-15-95	n.a.	**6689421**	n.a.	6773876	n.a
40-07-75	7683214	**7405793**	3.75%	**7405793**	3.75%
40-09-95	8184249	**7667719**	6.74%	7747974	5.63%
40-10-65	9482857	**7265559**	30.52%	**7265559**	30.52%
50-10-65	10785572	**10513029**	2.59%	**10513029**	2.59%
50-10-75	12073994	**11217503**	7.64%	11282024	7.02%
50-10-95	15820477	**12845598**	23.16%	12998064	21.71%
Average	*6031979*	*5534334*	*8.99%*	*5551066*	*8.66%*

Table 4: Comparison between the solution values of the memetic heuristic (M) and the CPLEX (C) branch-and-bound algorithm. Bold entries correspond to the minimum solution value for each row.

	Solution value			Computation time (sec.)		
Instance code	Genetic	Tabu	Memetic	Genetic	Tabu	Memetic
10-5-E-G	1043400	**1021050**	**1021050**	1	1	2
10-5-E-H	1229000	**1188500**	**1188500**	2	1	2
10-5-F-G	**1433550**	**1433550**	**1433550**	4	3	2
10-5-F-H	**1471950**	**1471950**	**1471950**	2	1	2
12-3-E-G	**1323900**	**1323900**	**1323900**	1	1	1
12-3-E-H	1953250	1828750	**1771450**	1	1	2
12-3-F-G	1915250	**1788750**	**1788750**	1	1	2
12-3-F-H	2094950	2002000	**1970500**	1	1	2
12-4-E-G	**1500150**	**1500150**	**1500150**	2	1	5
12-4-E-H	**2081370**	**2081370**	**2081370**	3	2	2
12-4-F-G	**1770500**	1776800	**1770500**	2	1	2
12-4-F-H	**2213500**	**2213500**	**2213500**	5	6	2
12-5-E-G	1580470	**1519100**	**1519100**	2	2	2
12-5-E-H	**2086100**	**2086100**	**2086100**	7	4	2
12-6-E-G	**1960750**	**1960750**	**1960750**	7	5	3
12-8-E-G	1263350	1259350	**1244300**	2	1	4
14-6-E-G	**2351000**	**2351000**	**2351000**	7	5	5
14-7-E-G	**2365650**	**2365650**	**2365650**	9	9	5
14-8-E-G	**2622470**	2665170	**2622470**	11	11	6
14-9-E-G	**2326370**	2373320	**2326370**	4	2	7
15-6-E-G	**2707850**	2718250	**2707850**	13	13	5
15-7-E-G	**2720070**	**2720070**	**2720070**	9	10	6
15-8-E-G	**2856850**	2913420	**2856850**	22	25	7
16-6-E-G	**2709300**	**2709300**	**2709300**	11	10	6
16-7-E-G	**2809870**	2919000	**2809870**	30	39	7
5-24-E-G	**381000**	**381000**	381000	2	1	2
5-30-E-G	**357100**	**357100**	**357100**	3	2	2
Average	*1893666*	*1886254*	*1872331*	*6*	*6*	*4*

Table 5: Comparison between the results of the memetic heuristic and its component: genetic and POTS. Bold entries are optimal values.

Instance code	With TABUFIX (sec.)	Without TABUFIX (sec.)
12-3-F-H	2	>7200
12-4-F-H	2	827
12-5-E-H	2	253
15-8-E-G	7	2569
16-6-E-G	6	337
16-7-E-G	7	55
Average	*5*	*808*

Table 6: Comparison between the computation times of the memetic heuristic with and without TABUFIX.